Resource Management for BG/L

Andy Yoo

Lawrence Livermore National Laboratory

UCRL-PRES-200217

Outline

- Resource management plan for BG/L
- SLURM overview
- Structure of resource manager for BG/L
- Resource management strategy
  - Job scheduling
  - System partitioning (Allocation)
  - Fault management
- Job submission and execution
- Current development status and future work
- Concluding remarks

Resource management objectives for BG/L

- High scalability
- High system utilization
- Simplicity
- Flexibility
- Support to boost the performance of user applications

BG/L resource management strategy

- One logical partition - single job queue for entire system
- Space-sharing only - multiple physical partitions and one user per physical partition
- We will use SLURM, a new resource manager developed at LLNL, combined with low-level machine and task management infrastructure provided by IBM
- Key resource management algorithms for BG/L
  - Efficient node allocation
  - Innovative job scheduling strategy

What is Slurm? Unknown Facts

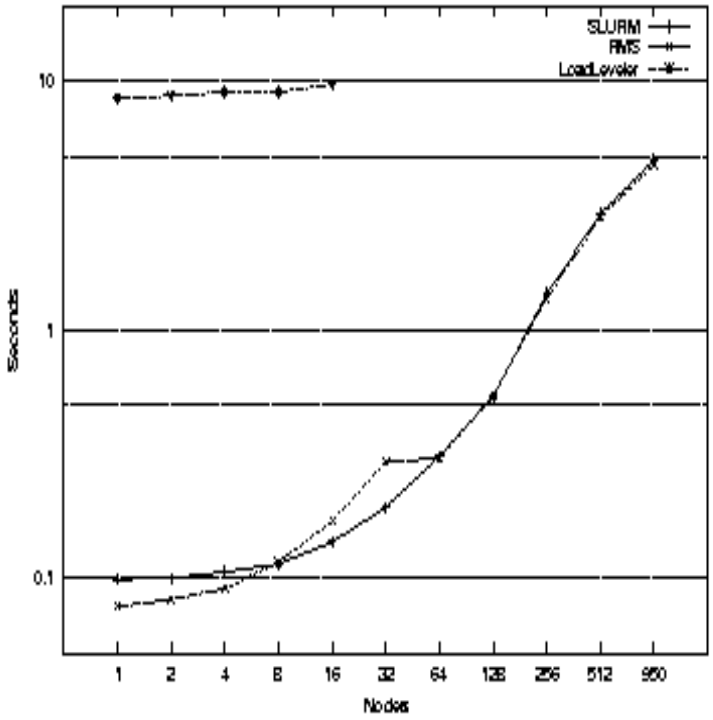
- Largest brand of carbonated beverage in the universe
- Manufactured on the planet Wormulon
- Rumor says it comes from the butt of a giant slug
- Slurm is *highly addictive*, but has so far not been conclusively linked to ailments

SLURM = simple, scalable, and flexible tool for resource management

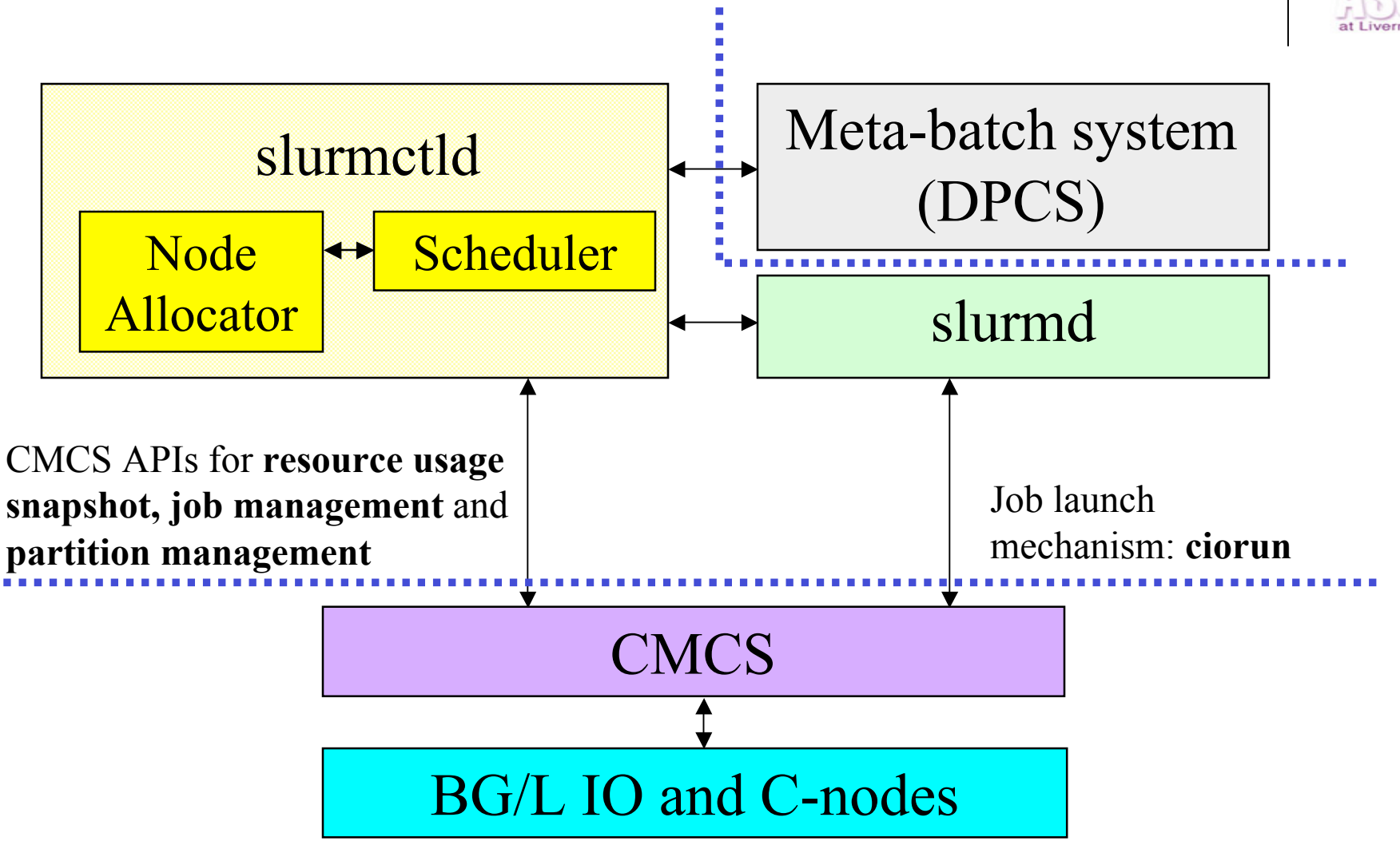
- What is SLURM?
  - Simple Linux Utility for Resource Management
  - Simple cluster manager that manages system resources and user jobs
  - Provides efficient and reliable execution environment for parallel jobs
  - Not a sophisticated job scheduler
  - Low-level scheduler for external meta-batch system
- Main SLURM design objectives – simple, scalable, flexible, fault-tolerant, secure, portable (open source)

The performance of SLURM

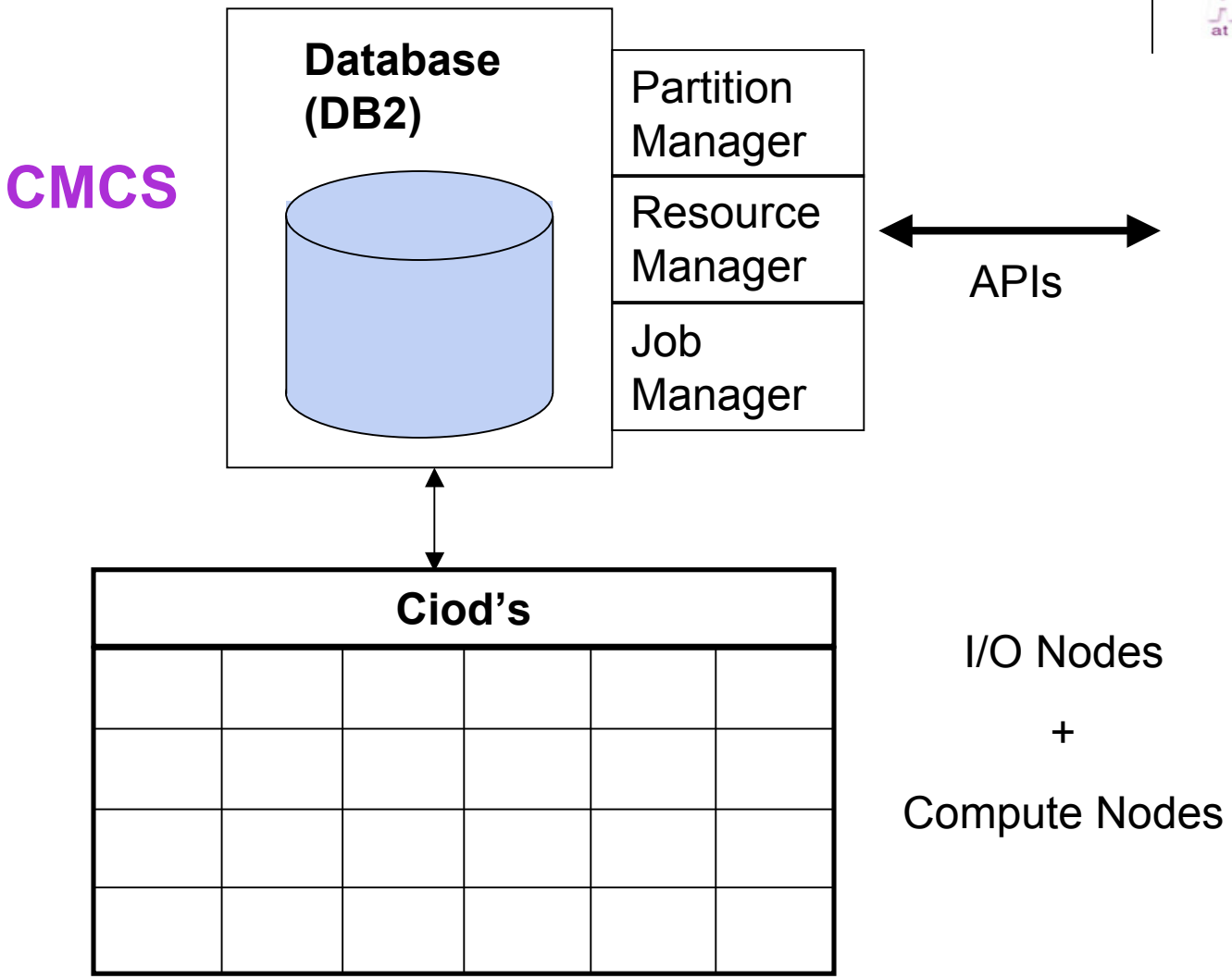
- Launched *hostname* on 950 nodes (1900 processors) of MCR cluster
- Launch performance
  - LoadLeveler: 9.8 sec (for 16 nodes)
  - RMS: 4.8 sec
  - SLURM: 4.7 sec
- SLURM will scale well on BG/L
  - Major components only have to scale to 1,024 IO nodes
  - Launch will be based on broadcast of binary from single IO node



Structure of resource management system for BG/L



Core Monitoring and Control System (CMCS)

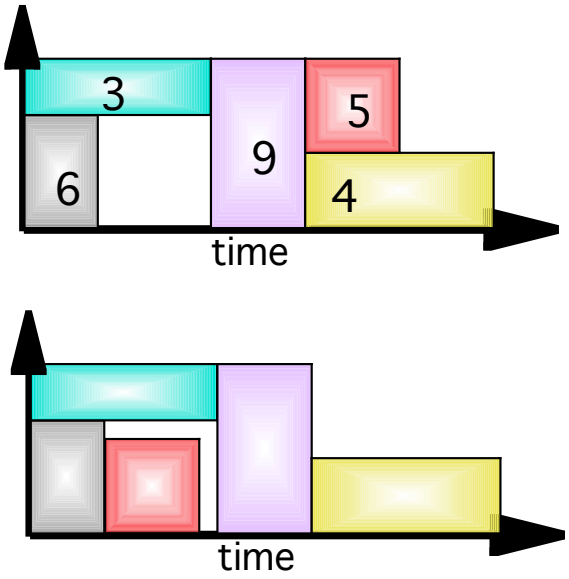


Job scheduling in BG/L

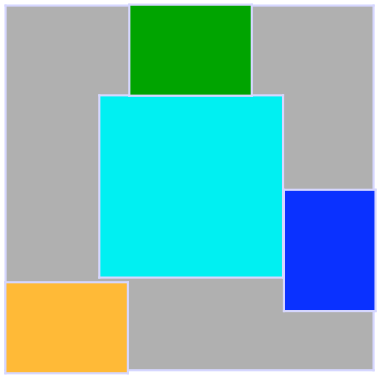
- Job scheduling determines when (*scheduling*) and where (*allocation*) to execute user jobs
- Job scheduling strategies can significantly impact the utilization of large computer systems
  - Machines like BG/L with toroidal topology (as opposed to all-to-all switch) are particularly sensitive
  - Utilization in the 50-70% range was observed in previous research
- Two scheduling techniques investigated for BG/L
  - Backfilling
  - Task migration

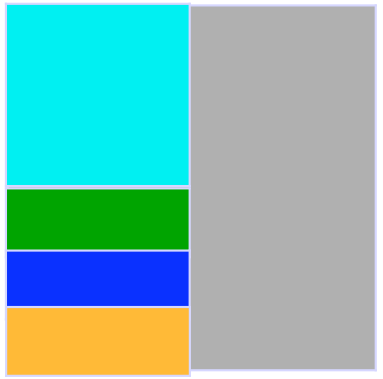
Backfilling and task migration

backfilling
queue = {6,3,9,5,4}, 10 nodes

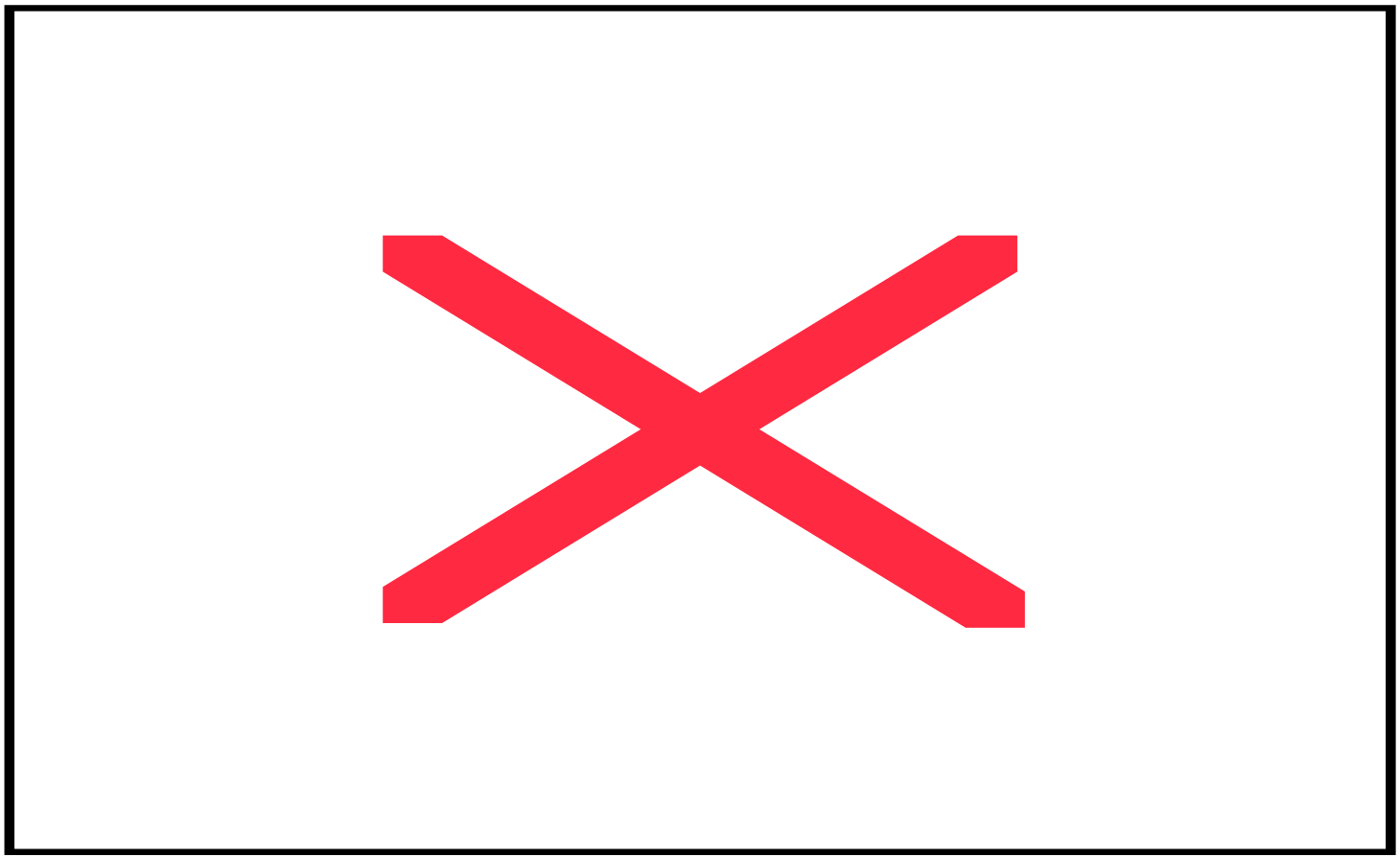


migration





Results for job scheduling on BG/L



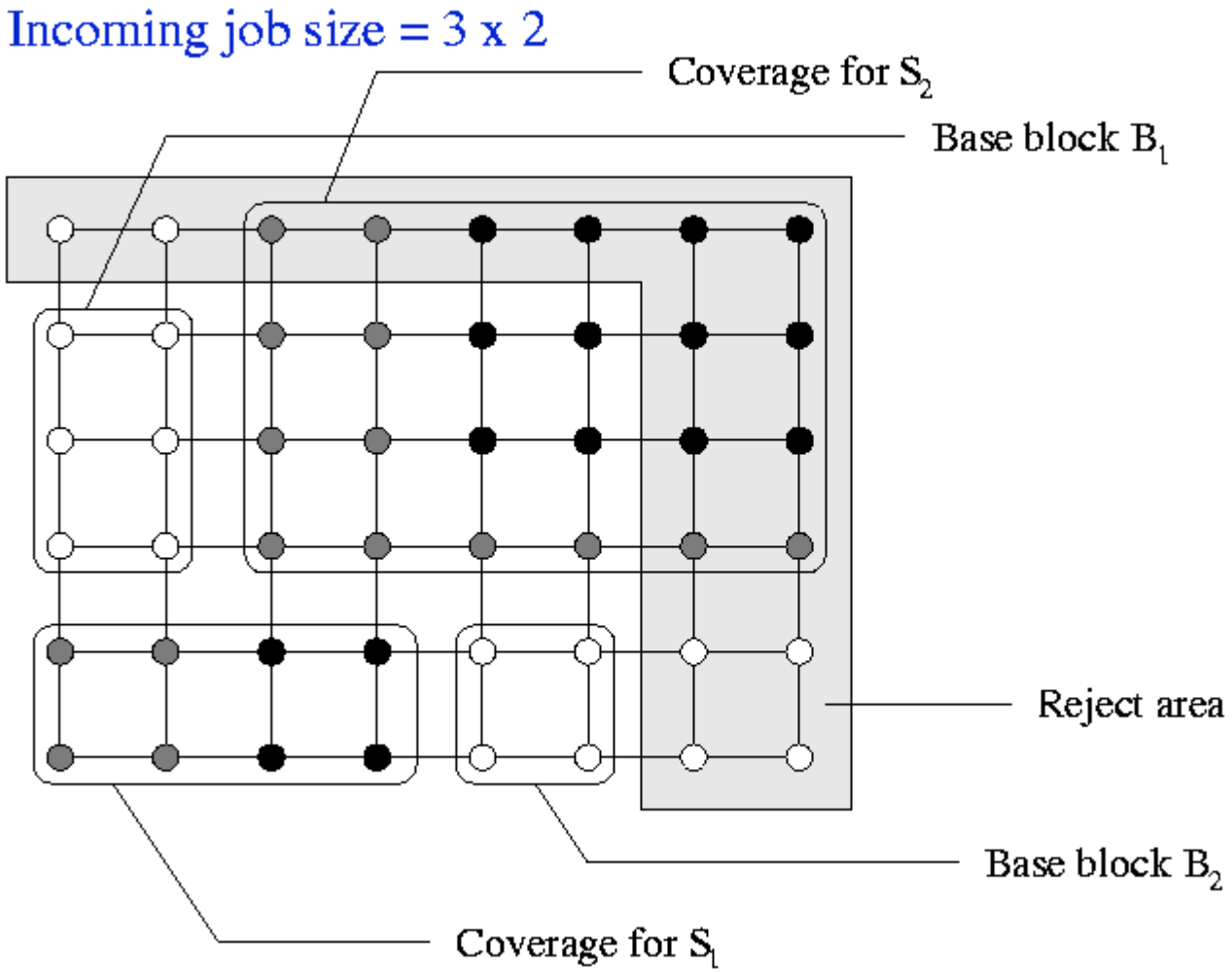
Job scheduling strategy for BG/L

- *Backfilling* technique will be used to improve machine utilization
- Scheduling of user jobs will be performed by the Distributed Production Control System (DPCS)
- The DPCS currently achieves high system utilization (90+%) with backfilling for the machines it manages
- Expect to achieve high system utilization and low job response time for BG/L

Allocation (or system partitioning) strategy

- Part of scheduling process
- Finds free compute nodes
- Partition management is handled by Core Monitoring and Control System (CMCS)
- Base partition (BP) – a basic scheduling unit (8x8x8 3D grid)
- A separate development/debug environment with smaller base partition

Allocation algorithm for 2D mesh



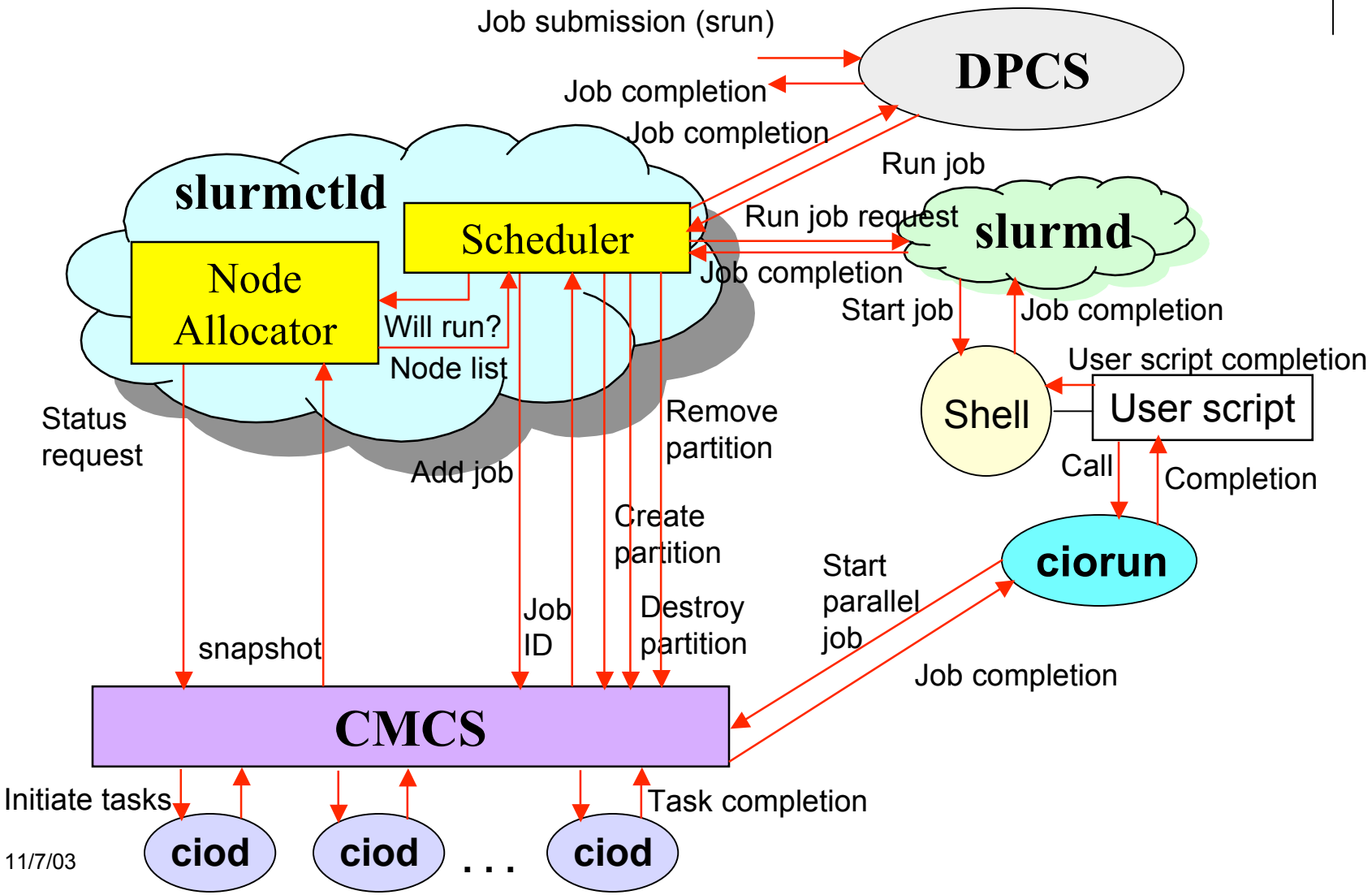
Processor allocation for BG/L

- An existing allocation algorithm for meshes will be used - first-fit, best-fit, worst-fit, and buddy system
- To be extended to handle 3D torus topology
- For BG/L, the availability of wires and switches needs to be checked to make sure all the nodes are connected

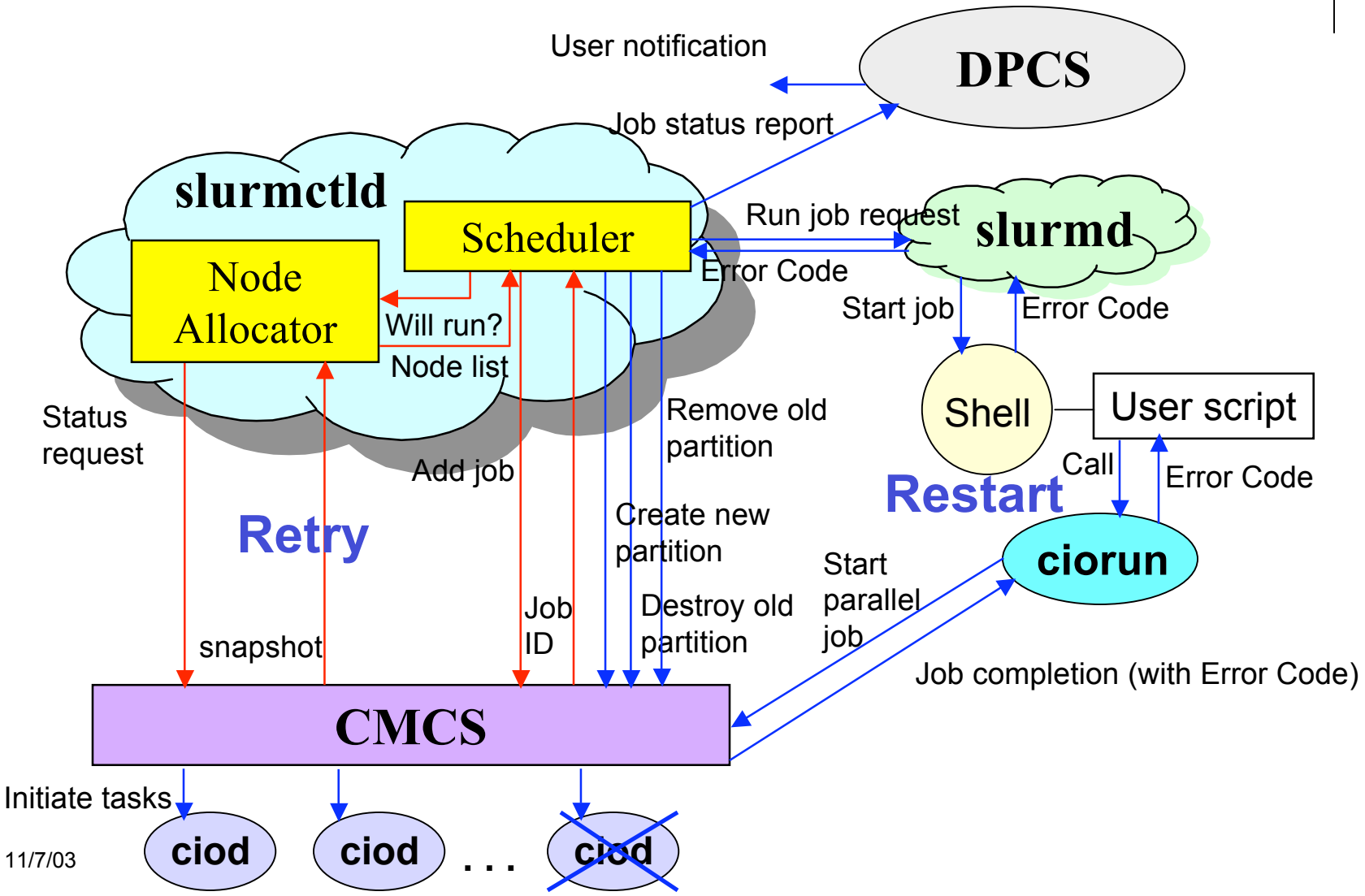
Fault Management in BG/L

- The components are monitored by control and monitoring subsystem and their status is recorded in the CMCS database (DB2)
- CMCS handles low-level fault management
- SLURM relies on the CMCS database to maintain machine status information
- Easy to checkpoint
- When a fault occurs, an attempt to restart the interrupted job will be made

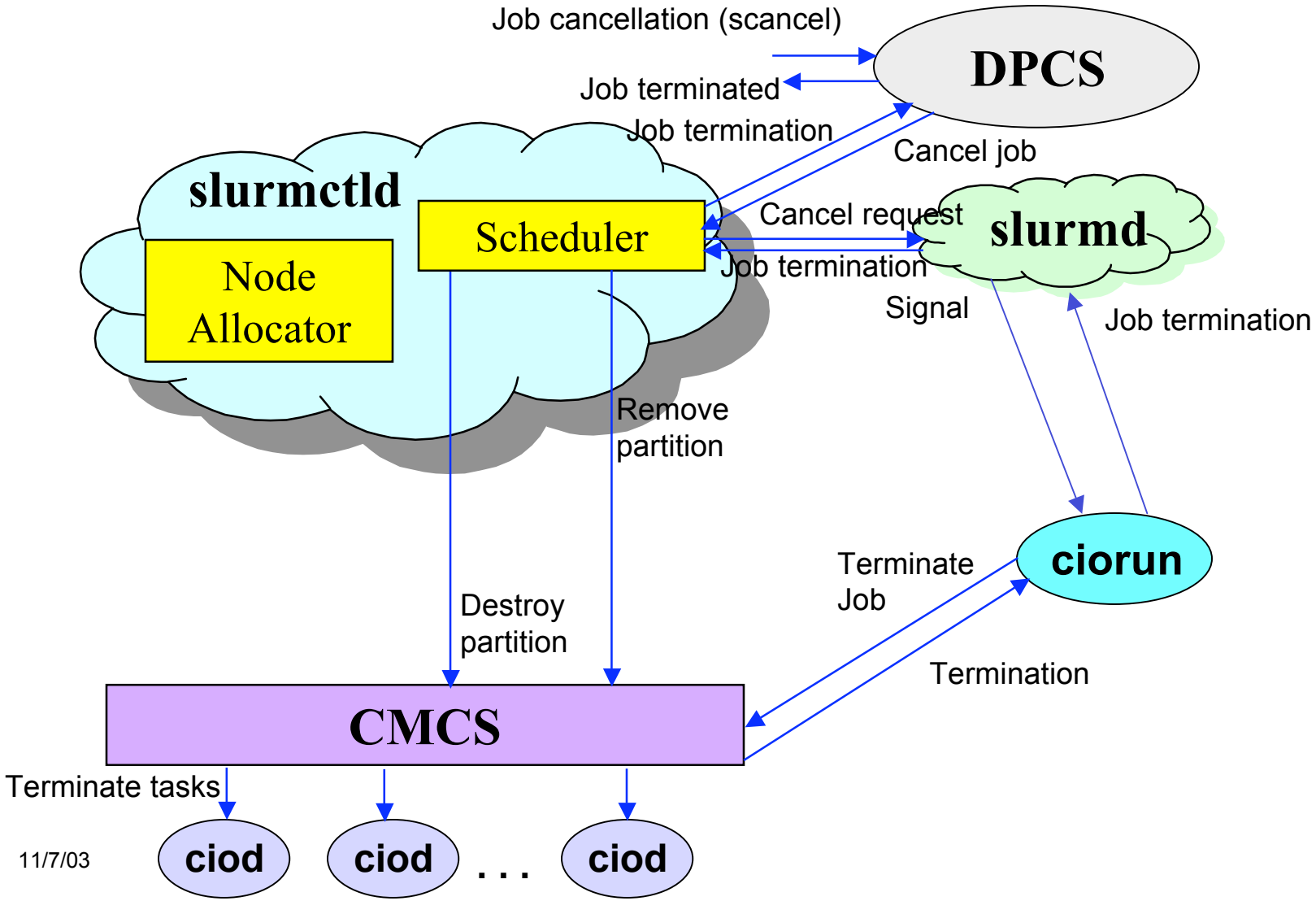
Normal execution of a job on BG/L



Abnormal termination of a job on BG/L



Cancellation of a job on BG/L



Current status of SLURM for BG/L development

- SLURM is currently operational on ASCI Linux Cluster (ALC)
- Design of SLURM for BG/L is complete
- Logic changes in SLURM and the APIs needed are identified.
- Currently in implementation phase
- Preliminary version of SLURM ported to BGLsim
- Successfully cross-compiled a “hello, world” program and ran via SLURM on the BGLsim under single- and multi-node configurations

Future work

- Design and implementation of allocation algorithm (on simulator)
- Development and evaluation of backfilling and other innovative scheduling schemes
- Interactive job handling

Conclusions

- Highly scalable resource management system
- Fast and reliable job management operations through SLURM and CMCS
- High system utilization through efficient scheduling and allocation techniques
- Simple and easy-to-use job execution environment

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Backup Slides

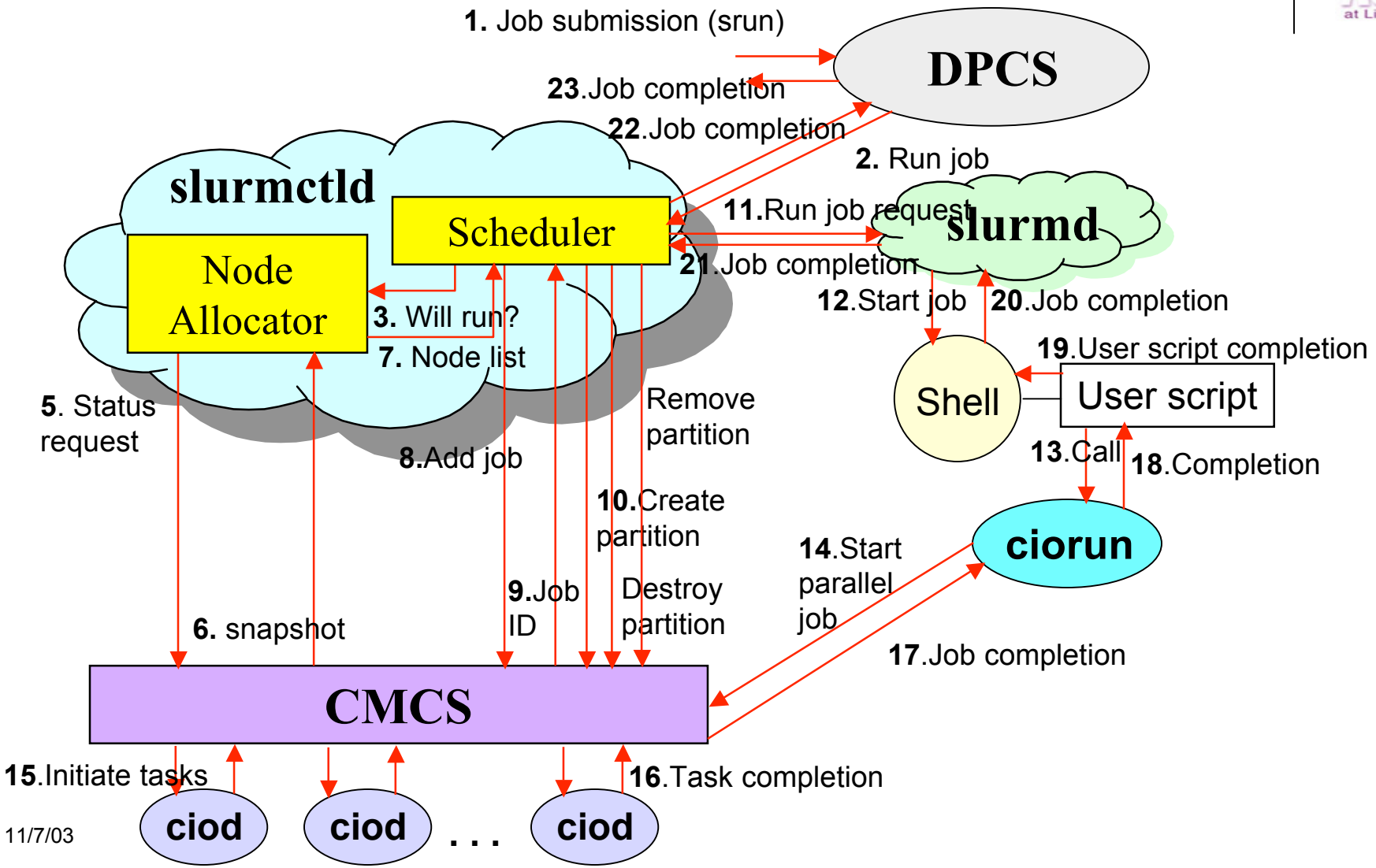
Allocation algorithm

1. Read current base partition status and construct coverages
2. Read the current state of wires and switches
3. While (there exists an unexploited coverage) do
    1. Find an available subsystem using allocation algorithm
    2. If (the subsystem meets connectivity test) then
        return the subsystem
       else
        mark the subsystem unavailable

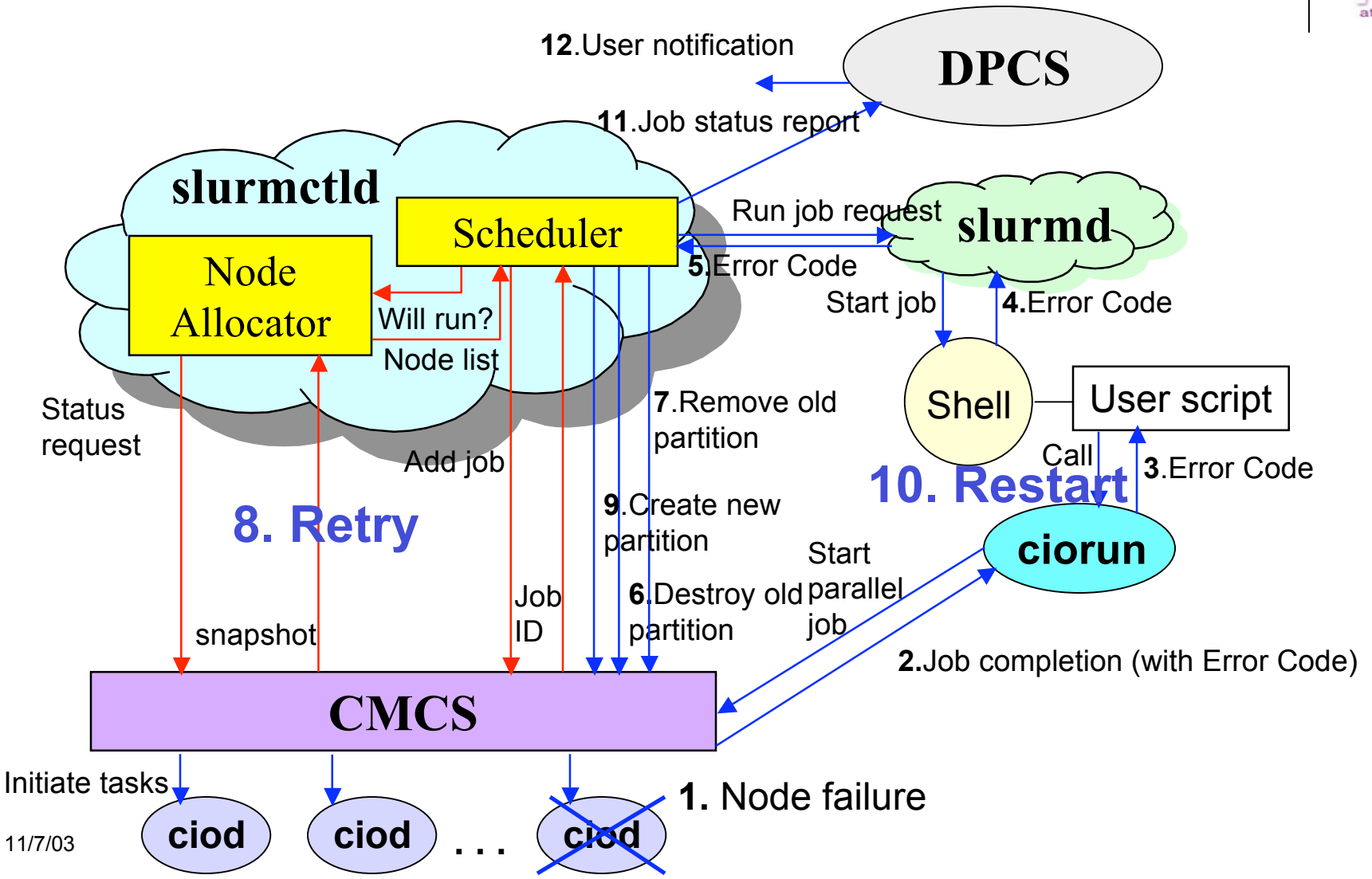
SLURM for BG/L functionalities

- Job scheduling and node allocation
- Machine/Job status monitoring
- Provides 5 simple commands for users: *srun, scancel, squeue, sinfo, scontrol*
- Allows users to specify the size and type of job partition
- Supports task-to-processor mapping for application performance improvement

Normal execution of a job on BG/L



Abnormal termination of a job on BG/L



Cancellation of a job on BG/L